electronics

MDPI

Article

A Computation Offloading Strategy in LEO Constellation Edge Cloud Network

Feihu Dong [1,2], Tao Huang [1,*], Yasheng Zhang [2], Chenhua Sun [2] and Chengcheng Li [2]

[1] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; dongfh@bupt.edu.cn
[2] The 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang 050081, China; zys163@163.com (Y.Z.); sun_chenhua@sina.com (C.S.); 931138419@163.com (C.L.)
* Correspondence: htao@bupt.edu.cn

Abstract: With the rise of a new generation of low Earth orbit (LEO) satellite constellations and the advancement of the 6G network, satellite–terrestrial integrated Internet of Things (IoT) in the future will achieve global coverage through the integration of LEO constellation, and extend computing to the edge of the network through the deployment of edge computing services in LEO constellation so as to meet the demand of mass connection and low latency data processing. The LEO constellation network of the future will be an edge cloud network combining network and computing. In this paper, we propose a computation offloading strategy for the combined optimization of energy and computational load in a LEO constellation edge cloud network (hereinafter referred to as LEO-ECN). First, we establish the LEO-ECN system model, in which the user task can be offloaded to the satellite through the multi-hop path. Then, a cost model considering energy consumption and load calculation is proposed. Finally, a joint optimization problem to minimize energy consumption and balance the LEO-ECN load is established, which is a convex optimization problem. The simulation result demonstrates that, compared with the benchmark strategy, our proposed strategy has better performance and can improve the computing resource utilization of LEO-ECN.

Keywords: LEO constellation; edge computing; computation offloading; joint optimization

Citation: Dong, F.; Huang, T.; Zhang, Y.; Sun, C.; Li, C. A Computation Offloading Strategy in LEO Constellation Edge Cloud Network. *Electronics* **2022**, *11*, 2024. https://doi.org/10.3390/electronics11132024

Academic Editor: Mehdi Sookhak

Received: 18 May 2022
Accepted: 24 June 2022
Published: 28 June 2022

1. Introduction

With the development of space technology, the cost of satellite manufacturing and launch has been greatly reduced, and the second wave of low-orbit constellation construction has arrived [1]. New space companies, such as OneWeb, SpaceX, Telsat and others, are ambitious to build low-orbit mega-constellations. SpaceX has laughed 1892 satellites [2] in orbit as of November 2021. A large number of LEO satellites run in space and form the satellite internet through inter-satellite links. LEO satellites act as data routing and forwarding nodes, but most nodes cannot effectively cover users on land most of the time, resulting in a waste of satellite resources and energy [3].

Along with the commercialization of 5G technology, academia has started research on 6G technology. The 6G technology is still in the stage of demand analysis and concept demonstration, but a consensus can be reached that the 6G network will be a ubiquitous network integrating the sky and the ground, connecting everything. However, the ground network cannot achieve global coverage, and it is impossible to lay optical fibers or set up base stations in deep mountains, seas, and deserts. At present, ground networks can only cover 20% of the global land area, and they are mainly concentrated in urban areas [4]. On the other hand, the ground network is very fragile and vulnerable to damage caused by various natural disasters. The low-orbit constellation communication has the advantages of seamless global coverage, ultra-low latency, and no need for ground infrastructure. It will be an indispensable and important part of Internet of Things and has become a research

hotspot in the industry and academia. Figure 1 shows the scenario of the satellite–terrestrial integrated Internet of Things.

The rapid development of the mobile internet has brought us many innovative user applications, such as voice/face recognition, 4K/8K high-definition video, VR/AR games, holographic communication [5], etc. Most of these new applications are computationally sensitive, which drives the development and deployment of edge computing in terrestrial Internet of Things.

In the coming years of satellite–terrestrial integrated Internet of Things, the network will extend to space. As a way for users to access Internet of Things, LEO satellites are at the edge of the entire network. Deploying edge computing servers on LEO constellation satellites to provide users with space edge cloud computing services is a matter worth considering. First of all, LEO satellites are at the edge of the network and can provide lower response delays. Even if users can access the data center through the ground network, the current global distribution of data centers is not uniform. In the case of long distances, the edge of LEO computing services can provide users with lower propagation delay [3]. Secondly, the LEO edge cloud does not need the support of ground infrastructure. On the one hand, it can operate autonomously and has strong damage resistance; on the other hand, it will not be constrained by geography, politics, economy and other factors, such as ground deployment. It is the only means for users without ground infrastructure to obtain nearby computing resources. Finally, the advantages of LEO constellation global coverage can lift the time and space constraints of users. Users anywhere in the world can access the edge of cloud computing services at any time. In the LEO constellation edge cloud, the LEO satellite is no longer only a user access and routing switching node, but also a node capable of providing computing services. The LEO constellation is no longer just a communication carrier network or pipeline, but a space-based computing network. Using the mega-constellation network to provide users with edge cloud computing services has the following benefits.

(1) It breaks through the capability limit of a single LEO satellite and can provide more powerful computing capacity.

(2) It avoids the waste of satellite resources and energy, and improves the cost–efficiency ratio of the LEO megaconstellation.

(3) The space cloud computing infrastructure has been built with independent operation and elastic destruction resistance, which is not limited by natural disasters and geographical environment.

(4) Provides edge computing services seamlessly and anywhere in the world.

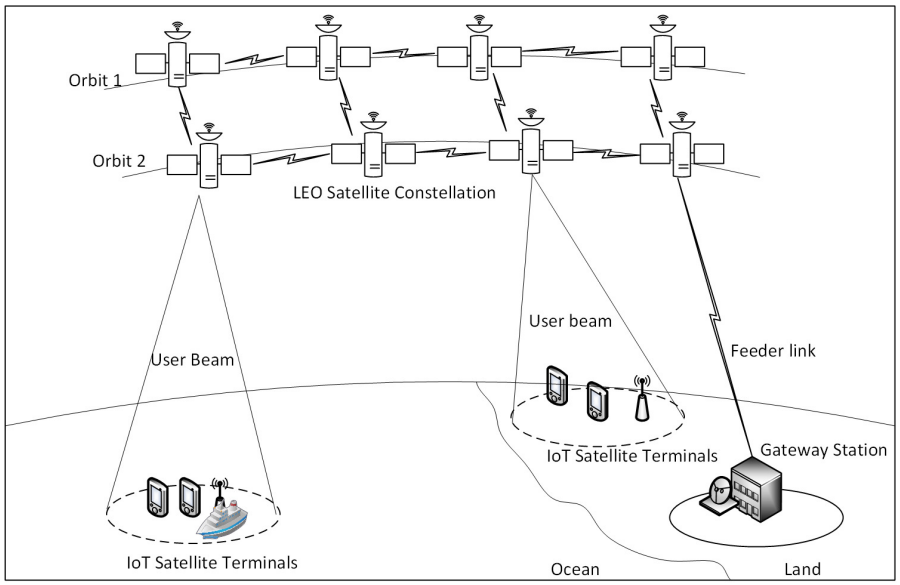

Figure 1. The satellite–terrestrial integrated Internet of Things scenario.

However, mega LEO constellation edge cloud network is a resource-constrained system, in which energy supply and computing capacity are much lower than the ground edge computing cloud. In order to improve the user capacity that the entire edge cloud can serve, the utilization of system resources needs to be optimized. On the one hand, the energy consumption of the system when the user offloads the task needs to be reduced; on the other hand, the computing resource utilization of the satellite in the LEO edge cloud needs to be balanced to avoid local congestion and overload. To the best of our knowledge, there are no studies that provide a task offloading strategy for edge computing cloud across the entire LEO constellation network.

In this work, we propose a joint optimization strategy of energy consumption and computing load for the LEO constellation edge cloud processing offloading task of user. Our main contributions are summarized as follows.

(1) The system model of LEO constellation edge cloud network is established. The model dispatches the computing resources of all satellites in the constellation to perform computing offloading for user tasks. Unlike other satellite edge computing models, the satellite terminal only considers offloading tasks to directly connected or neighbor satellites.

(2) The influence of the network path on computational offloading performance is considered, and the energy consumption caused by the inter-satellite multi-hop path is taken as part of the system cost. However, the influence of the path on computational offloading is often ignored in the existing literature.

(3) A cost model for combined optimization of energy consumption and computational load is proposed. Under the condition of limited computing resources in the low-orbit constellation, the system performance of the entire constellation edge cloud network is optimized by minimizing the total energy consumption of all users and balancing the computing load of all satellites in the constellation. The numerical simulation results show that the proposed strategy is superior to the benchmark strategy.

The remainder of this paper is organized as follows. In Section 2, an overview of the existing work is given. Section 3 describes the system model of the LEO constellation edge cloud network. The joint optimization problem formulation and the operation mechanism are presented in Section 4. Section 5 shows the simulation results with the proposed strategy. Finally, this work is concluded in Section 6.

2. Related Works

At present, there has been some research on LEO satellite edge computation, mainly focusing on how the computation task makes the offloading decision between the satellite and satellite terminals. For example, in the multi-user single-satellite access scenario, Ref. [6] established the LEO computation offloading model based on the joint optimization of system energy consumption and user delay and used the Nash equilibrium method of game theory and the Lagrange multiplier method of convex optimization to solve the optimal offloading strategy, step by step. In the multi-user and multi-satellite access scenario [7] proposed a two-layer computing offloading scheme integrating the satellite and the satellite terminal. The problem was formulated to an allocation problem, and the improved Hungarian algorithm was used to solve the problem. Simulation analysis was carried out from the two dimensions of reducing the task average delay and system energy consumption; Ref. [8] proposed a three layer architecture for multi-user multi-satellite computing offloading scheme, including terrestrial users, satellite and ground cloud services, which considers single satellite coverage time and limited in-orbit computing power. The offloading decision was solved from the perspective of minimizing the total energy consumption of ground users. First, by adopting a relaxation method, the nonconvex problem is converted into a linear programming problem. Then, a distributed solution for large-scale networking is proposed based on the ADMM method. In [4], considering the dynamic characteristics of the satellite, the communication time between offloading satellite and ground user was limited, the model for the ground user and satellite was established based on queuing theory, taking the average response time and average energy

consumption of a single task as optimization objectives, and the only optimal offloading strategy was obtained based on the Nash equilibrium in game theory.

In the above literature, the LEO edge computation offloading strategy mainly uses the access satellite or access satellites' neighbors. Although the over-top satellite can provide the lowest delay, due to the limitations of satellite-carried devices in a space environment, volume, weight, energy and other aspects, the processing capacity of a single satellite is very limited, unable to serve enough users, resulting in the number of users being limited. On the other hand, due to the uneven distribution of users, overtop satellites are used in user-dense areas, which results in too few satellite computing resources allocated and an increase in the response delay of a single user. The researchers gradually realized the possibility of using the LEO constellation network to provide edge computing services and developed some preliminary ideas. In [5], a satellite-ground integrated edge computing network architecture (STECN) was proposed, in which the LEO satellite network deployed on the MEC platform was included, and its system architecture, collaborative offloading process, functional composition, and challenges and problems were described in detail. Ref. [3] discussed the possibilities and challenges of providing satellite-borne cloud computing services on a new generation of mega constellations, and discussed three application cases that can benefit from them, one of which is edge computing. In [9], the edge computing architecture of low-orbit constellation communication network was designed, but only one satellite orbit was considered in the modeling, and the offloading range was limited to the satellite within two hops from the user. A summary of the literature is listed in Table 1.

Inspired by the above facts, this paper proposes an edge computing resource allocation strategy in a mega LEO constellation, which considers the unified arrangement of the energy resources and computing resources of the entire LEO constellation. A shared LEO constellation resource pool is formed for dynamic use by all users. Like the data center on the ground, all servers in the data center are scheduled and managed in a unified way to provide elastic and scalable computing resources.

Table 1. Summary of literature.

References	Offloading Scenario	Optimization Goal	Full Constellation Offload
[3]	-	-	mentioned
[4]	multi-user single-satellite	user response time & user energy	no
[5]	-	-	mentioned
[6]	multi-user single-satellite	system energy & user delay	no
[7]	multi-user multi-satellite	system energy & user delay	no
[8]	multi-user multi-satellite	user energy	no
[9]	multi-user one satellite orbit	-	mentioned

3. System Model

In this section, the system model is shown in Figure 2, which is composed of LEO satellites, satellite terminals, gateway stations and the edge computing orchestrator. Each LEO satellite is equipped with router units and onboard MEC servers. The router unit is used for inter-satellite data exchange, and the onboard MEC server provides edge computing services. The router unit of each satellite node is connected with the intra-plane and inter-plane links to form an inter-satellite routing and forwarding network and realize the network layer reachable between MEC servers. As the control node of the edge cloud

network, the edge computing orchestrator is deployed behind the ground gateway station, which receives the computing request of the satellite terminal and centrally schedules user tasks to be offloaded by the satellite terminal among LEO satellites.

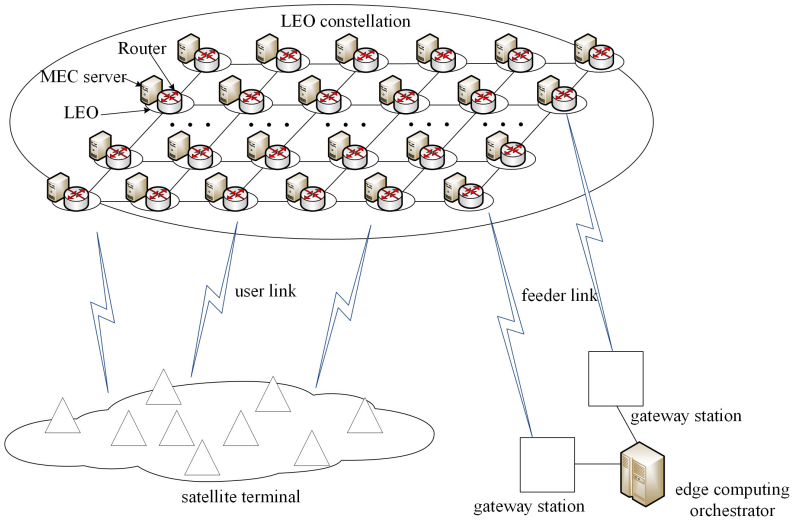

Figure 2. The LEO-ECN scenario.

In this model, there are M LEO satellites. The single satellite is denoted as m. The set of LEO satellites can be denoted as $\mathcal{M} = \{m \in \mathbb{N}^+ | m \in [1, M]\}$. There are N satellite terminals, the terminal is denoted as n, the set of satellite terminals can be denoted as $\mathcal{N} = \{n \in \mathbb{N}^+ | n \in [1, N]\}$. The notations used in this paper are summarized in Table 2. Next, we introduce the communication, network, computation, load models in detail.

3.1. Compute Offloading Mechanism

In LEO-ECN, the edge computing orchestrator has the global view of the entire LEO cloud network system, including the network resource view and computing resource view. The network resource view contains topology and routing information of the entire system, which changes regularly and predictably with the relative motion of satellites. The computing resource view contains the computing resource usage and remaining status of all satellites, which changes with the previous and current amounts of offloading tasks. The orchestrator, as the control center of the entire system, can obtain the latest global status information in real time.

When the user needs to offload the computing task, the satellite terminal will send the computing offloading request to the orchestrator. The orchestrator summarizes all computing requests $\mathcal{W}$ at the current time t and runs the optimal algorithm combining with the network resources and computing resources at the current moment to make the optimal offloading decision, determine the satellite nodes for each user to provide computing service, which will be announced to each satellite terminal, and then the satellite terminal sends the task data to the service satellite arranged by the orchestrator. After the calculation is completed, the service satellite sends response data to the satellite terminal, and finally completes the task of offloading. The main process is shown in Figure 3.

Considering the autonomous operation of the whole system and the reduction in signaling delay, the orchestrator, as a control node, also can be distributed on satellite nodes with high performance, and status information can be synchronized between control nodes through distributed consistency protocol. On the other hand, compared with virtual machines, containers can make more efficient use of the limited computing resources of the space-borne MEC server, and computing services are provided by enabling the space-borne MEC server to instantiate containers.

Table 2. Notation definitions.

Symbol	Definition
$\mathcal{M}$	Set of LEO satellite m
$\mathcal{N}$	Set of ground terminals
$\mathcal{W}$	Set of computation task
D_n	Data size of task n
C_n	Number of CPU cycles for task n
B_n	Channel bandwidth of ground terminal n
G_t	Transmitting antenna gain of ground terminal n
G_r	Receiving antenna gain of ground terminal n
L_p^{gs}	Rain attenuation of ground-satellite link
L_n^{gs}	Free space loss of ground terminal n
N^{gs}	Noise power of ground terminal to access satellite
P_n^{gs}	Transmit power of ground terminal n
$R_{i,j}^{ss}$	Inter-satellite link transmit rate from satellite i to j
K_b	Boltzmann's constraint
T_s	System noise temperature
E_b/N_0	Ratio of received energy-per-bit to noise power density
$P_{i,j}^{ss}$	Transmit power of satellite i to satellite j
$L_{i,j}^{ss}$	Free space loss of the inter-satellite link
$G(k)$	Graph of the k topology snapshots
V	Set of satellite nodes in the graph $G(k)$
$E(k)$	Set of inter-satellite links in the k snapshot
F_m	Maximum CPU cycles provided by onboard MEC server
$E_{n,m}^{p}$	Energy consumed on the path from ground terminal n to service satellite m
$E_{n,m}^{c}$	Energy that service satellite n consumes for task n
hop	Number of hops ground terminal to service satellite
f_m	CPU frequency of the MEC server on the satellite m
Θ	The total cost of compute and path

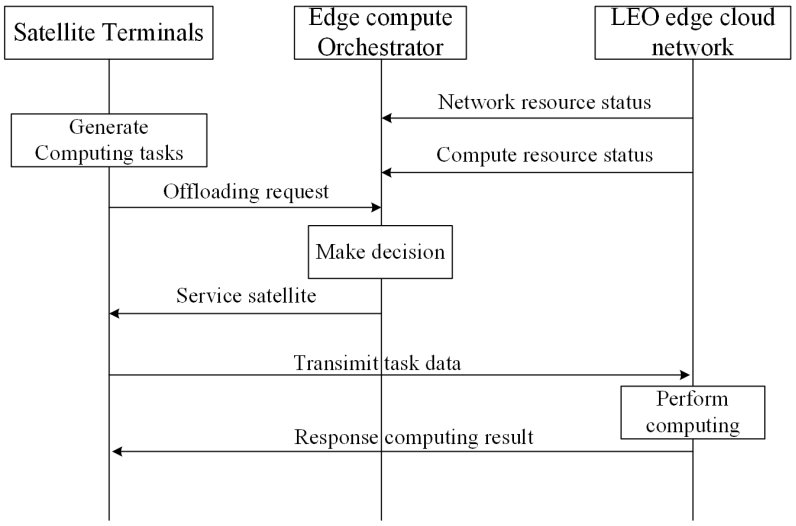

Figure 3. Compute offloading process.

3.2. Communication Model

We assume that each satellite terminal uses TDMA multiple access and can access only one satellite at a time. Benefiting from the global coverage of low-orbit constellations, satellite terminals can always be covered by low-orbit satellites. The satellite–ground link

and the inter-satellite link use microwaves. Satellite terminals uplink transmit rate R_n^{gs} can be expressed as [10]:

$$R_n^{gs} = B_n \log_2 \left(1 + \frac{P_n^{gs} G_r G_t L_p^{gs} L_n^{gs}}{N^{gs}} \right), \forall n \in \mathcal{N} \tag{1}$$

where B_n denotes the channel bandwidth of satellite terminal n. G_t and G_r stand for the transmitting antenna gain and the receiving antenna gain, respectively. L_p^{gs} denotes the rain attenuation. L_n^{gs} is the free space loss. N^{gs} denotes the noise power from ground to access satellite. P_n^{gs} denotes transmit power of satellite terminal n. The inter-satellite link transmit rate $R_{i,j}^{ss}$ is computed as [11]

$$R_{i,j}^{ss} = \frac{P_{i,j}^{ss} G_r G_t L_{i,j}^{ss}}{K_b T_s (E_b / N_0)}, \forall i, j \in \mathcal{M} \tag{2}$$

where K_b is Boltzmann's constant, T_s is the system noise temperature, and (E_b/N_0) is the ratio of received energy-per-bit to noise power density. $P_{i,j}^{ss}$ denotes the transmit power of satellite i when data are sent to satellite j. $L_{i,j}^{ss}$ is the free space loss of the inter-satellite link. We assume the transmitted power P_n^{gs} and $P_{i,j}^{ss}$ are fixed values, and thus the energy consumption is proportional to the time transmitting data from satellite terminal to satellite.

3.3. Network Model

Different from the fixed topology of the terrestrial mobile network and wired network, the LEO constellation network has a dynamic time-varying discrete topology as illustrated in Figure 4. The dynamic network topology is considered a periodically repeating series of K topology snapshots [12] separated by every inter-satellite link switching. Each of the snapshots is modeled as a graph $G(k) = (V, E(k))$, where V is the set of satellite nodes, and $E(k)$ is the set of inter-satellite links in the k snapshot.

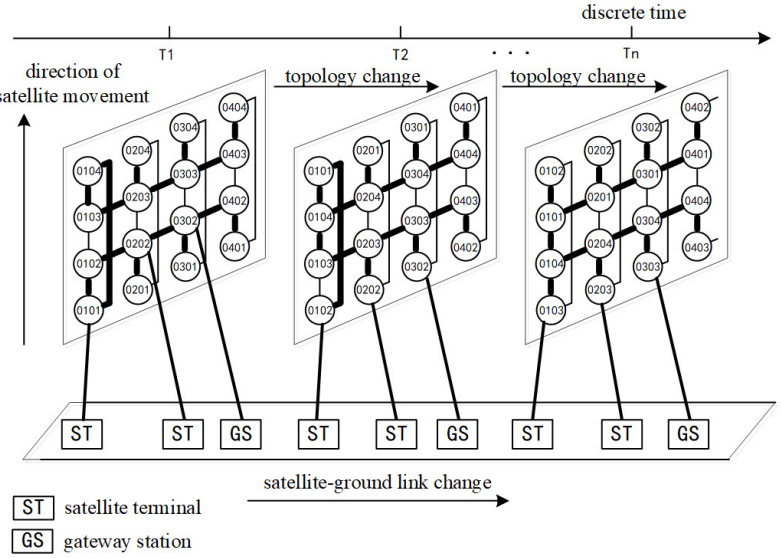

Figure 4. The LEO constellation network discrete topology.

In the LEO constellation edge cloud network, the computing data of the users may reach the satellite providing computing offloading service through inter-satellite multi-hop path after being scheduled by the orchestrator. The cost of multi-hop path cannot be ignored because of the delay and energy consumption of each hop link. We assume that the ground terminal under each satellite transmits calculation data to the destination satellite using the shortest path. When the topology snapshot changes, the shortest path may also

change. So the Dijkstra algorithm runs to calculate the shortest path from each node to other nodes in each topology snapshot. The shortest path contains the next hop to other nodes and the number of hops to go through. When the satellite–ground link changes, for the continuous offloading tasks, such as video codec, the migration overhead caused by shortest path tree switching should be considered. However, it has little influence on the second-level real-time response tasks. The offloading task model studied in this paper is the second-level response task.

3.4. Computation Model

For the sake of simplicity, each satellite terminal generates only one computation task at a time, and the set of tasks is denoted as $\mathcal{W} = \{W_n | W_n \triangleq \{D_n, C_n\}, n \in \mathcal{N}\}$. D_n represents the amount of data that the task W_n needs to upload in bytes. C_n represents the required CPU computing capacity in cycles per second. In space, onboard devices require additional protection from the space environment, especially from particle radiation. On the other hand, satellites are strictly limited in size, power consumption and weight. Therefore, the performance of spaceborne MEC servers is much worse than that of ground-based edge computing servers, and its computing capacity is very limited. Assume that the total computing capacity of each satellite is equal, and the maximum value is expressed as F_m in cycles per second. A MEC server on a single satellite may not be able to meet the computing requirement of a user's task. It is reasonable to assume that the computing task can be divided, and the variable $x_{n,m}$ is used to represent whether the computing task n of the ground terminal is offloaded to satellite m. Then, there are the following constraints:

$$\sum_{m \in \mathcal{M}} x_{n,m} = 1, \forall n \in \mathcal{N} \tag{3}$$

$$\sum_{n \in \mathcal{N}} x_{n,m} C_n \leq F_m, \forall m \in \mathcal{M} \tag{4}$$

$$x_{n,m} \in [0, 1] \tag{5}$$

4. Problem Formulation and Analysis

4.1. Energy Cost

4.1.1. Path Energy Cost

For the energy model, since user task data may reach the satellite providing computing service in LEO Constellation edge cloud network through multi-hop path, the energy consumption of a single user's computing task should include path energy consumption in addition to CPU computing energy consumption of onboard MEC servers. Path energy consumption refers to the energy consumption generated when the task data sent from the ground terminal to the service satellite pass through multi-hop forwarding nodes and the energy consumption generated when the response data sent from the service satellite to the ground terminal passes through multi-hop forwarding nodes. We call the former the forward path energy consumption expressed as $E_{n,m}^{fp}$ and the latter the reverse path energy consumption expressed as $E_{n,m}^{bp}$. Therefore, the total path energy consumption of a computation task is expressed as

$$E_{n,m}^{p} = E_{n,m}^{fp} + E_{n,m}^{bp} \tag{6}$$

For simplicity, assume that the forward and reverse paths are symmetric and consistent, i.e., the forward and reverse paths pass through the same number of nodes and hops. Forward and backward path energy consumption can be further expressed as

$$E_{n,m}^{fp} = E_{n,m}^{fp,1} + \sum_{i=2}^{hop} E_{n,m}^{fp,i} \tag{7}$$

$$E_{n,m}^{bp} = \sum_{i=1}^{hop-1} E_{n,m}^{bp,i} + E_{n,m}^{bp,hop} \tag{8}$$

where hop indicates the number of hops on the path, $E_{n,m}^{fp,1}$ represents the energy consumption of the first hop transmission in the forward path, namely, the energy consumption of the satellite terminal satellite–ground uplink, $E_{n,m}^{bp,hop}$ represents the energy consumption of the last hop of the reverse path, that is, the energy consumption of the satellite terminal satellite–ground downlink, $\sum_{i=2}^{hop} E_{n,m}^{fp,i}$, and $\sum_{i=1}^{hop-1} E_{n,m}^{bp,i}$ respectively represents the total energy consumption of inter-satellite multiple hops in the forward and reverse path. Considering that the amount of data after calculation is generally small, the transmission energy consumption of the reverse path can be ignored for simplicity. Therefore, the total energy consumption of the path can be modeled as

$$E_{n,m}^{p} = E_{n,m}^{fp,1} + \sum_{i=2}^{hop} E_{n,m}^{fp,i} \tag{9}$$

$$E_{n,m}^{fp,1} = P_n^{gs} \frac{x_{n,m} D_n}{R_n^{gs}} \tag{10}$$

$$E_{n,m}^{fp,i} = P_i^{ss} \frac{x_{n,m} D_n}{R_i^{ss}} \tag{11}$$

4.1.2. Computation Energy Cost

The computation energy consumption denoted as $E_{n,m}^{c}$ refers to the amount of energy that service satellites m consume to perform a satellite terminal's computing task n. The computational requirement for tasks W_n assigned to satellite m is $x_{n,m}C_n$, and the computation data volume assigned to satellite m is $x_{n,m}D_n$. According to the realistic measurements [13], the computation energy consumption is proportional to the square of the CPU frequency, and specifically $E_{n,m}^{c}$ is described as

$$E_{n,m}^{c} = \varepsilon (f_m)^2 x_{n,m} C_n \tag{12}$$

where ε is the energy coefficient which depends on the chip architecture. Without loss of generality, we think the CPU frequency f_m is different on the satellites due to the heterogeneity of the onboard MEC server.

Therefore, the total energy consumption of service satellite m for the computing task of ground terminal n can be expressed as

$$\begin{aligned} E_{n,m} =& E_{n,m}^{p} + E_{n,m}^{c} \\ =& P_n \frac{x_{n,m} D_n}{R_n} + \sum_{i=1}^{hop} P_i \frac{x_{n,m} D_n}{R_i} \\ &+ \varepsilon (f_m)^2 x_{n,m} C_n \end{aligned} \tag{13}$$

The total energy consumption of the computation task n offloading to the LEO constellation edge cloud can be expressed as

$$\sum_{m \in M} E_{n,m} \tag{14}$$

4.2. Computing Load Cost

In the LEO-ECN, LEO satellites cannot effectively cover users most of the time, resulting in idling and waste of resources. In addition, due to the uneven distribution of ground terminals and user tasks, unloading tasks from ground terminals to directly connected satellites is easy to cause local overload of computing resources on satellites. From the

perspective of system, tasks can be balanced to those idle satellites as much as possible so as to improve the efficiency and resource utilization of the whole low-orbit constellation edge cloud. We sum up the mean square error of the computing power demand assigned to each satellite as the cost of load balancing in a low-orbit constellation edge cloud network. The cost is related to the level of load balancing. Specifically, the more balanced the load, the smaller the cost. The system load balancing cost can be expressed as

$$\sum_{m\in\mathcal{M}}(\sum_{n\in\mathcal{N}}C_n x_{n,m}-\sum_{n\in\mathcal{N}}C_n/M)^2 \tag{15}$$

4.3. Objective Function

From the perspective of the system, the LEO-ECN has limited energy resources, so it needs to minimize the energy consumption of all tasks, and on the other hand, the computing capacity is also limited, and we should properly allocate the computing resources of all satellites, balance system load and improve system resource utilization [5]. Therefore, the overall cost function could be formulated as Equation (16).

$$\begin{aligned}\min_{n,m}\Theta &= \min_{n,m}\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}E_{n,m}+\beta\sum_{m\in\mathcal{M}}(\sum_{n\in\mathcal{N}}C_n x_{n,m}-\sum_{n\in\mathcal{N}}\frac{C_n}{M})^2\\ &= \min_{n,m}\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}}(P_n\frac{x_{n,m}D_n}{R_n}+\beta\sum_{i=2}^{hop}P_i\frac{x_{n,m}D_n}{R_i}+\varepsilon(f_m)^2x_{n,m}C_n)\\ &+\beta\sum_{m\in\mathcal{M}}(\sum_{n\in\mathcal{N}}C_n x_{n,m}-\sum_{n\in\mathcal{N}}\frac{C_n}{M})^2\end{aligned} \tag{16}$$

In (16), Θ denotes the overall cost, and β is the weight of the load balance cost. With the definition and the constraints of (3)–(5) above, the objective function is a quadratic programming problem, which is convex, since $x_{n,m}$ is continuous. We use opensource software CVXPY [14] to solve the problem.

5. Numerical Simulation

In this section, to verify the performance of our proposed strategy, we conducted a series of evaluation experiments. First, we simulated and compared the LEO constellation cloud network offloading strategy with the access satellite offloading strategy and neighbor satellite offloading strategy proposed in the existing papers. Then, we evaluated and compared our joint optimization strategy and benchmark strategy in the whole LEO constellation offloading scenario.

5.1. Simulation Setup

5.1.1. Constellation Parameter

We used open source software SAVI [15] to design three low-orbit satellite constellations with different numbers of satellites, and then obtained the snapshots of the three constellations at a certain time through ephemeris calculation, extracted the corresponding topological information according to the snapshots, which was used to calculate the shortest path between any satellites. Considering the batch and iterative update of satellite payload, the performance of MEC servers on each satellite is different. The average dominant frequency of MEC servers CPU is set to 1 GHz, and the standard deviation is 0.2. The energy coefficient of CPU ε is set to 10^{-28}. As the loads of each MEC server are also different at different time slots, the computation resources of MEC are assumed to be normally distributed with the mean of 2 GHz and standard deviation of 0.2. The parameters of the three low-orbit constellations are shown in Table 3.

Table 3. Constellation parameters.

Parameter	Constellation 1	Constellation 2	Constellation 3
constellation type	near-polar	near-polar	near-polar
number of orbits	6	6	12
satellites per orbit	6	12	12
inclination	86.4°	86.4°	86.4°
semimajor axis	7158.14 km	7158.14 km	7158.14 km
altitude	780 km	780 km	780 km
longtitude boundary	75°	75°	75°
CPU frequency (Ghz) per MEC server	$\mathcal{N}(1, 0.2)$	$\mathcal{N}(1, 0.2)$	$\mathcal{N}(1, 0.2)$
computation capacity per satellite (Gcycle/s)	$\mathcal{N}(2, 0.2)$	$\mathcal{N}(2, 0.2)$	$\mathcal{N}(2, 0.2)$

5.1.2. User Task Parameter

The user task distribution of each satellite is not balanced. There are many reasons that affect the user task distribution on the satellite. The first is geographical. Specifically, the majority of the population lives and produces on land, so when the satellite covers the ocean, the number of users is very small, and when the satellite passes over land, the number of users' tasks is increased. The second is the economic factor. Users in economically developed areas will have more tasks, because there are more netizens and more demands for computing and offloading. Third, due to the relative movement of the low-orbit constellation, the ground area connected to the same satellite will change, leading to changes in the distribution of user tasks on the satellite. For simplicity, we assume that the user tasks are mainly concentrated on satellites over the land, and the tasks are evenly offloaded to the current satellite nodes covering the land. As shown in Figures 5–7 [16], satellite red indicates high user task density. The amount of data D_i required to offload for each user task conforms to the normal distribution $\mathcal{N}(10, 2)$ Mbytes, and the computing resources C_i required conform to the normal distribution $\mathcal{N}(1, 0.2)$ Gcycles.

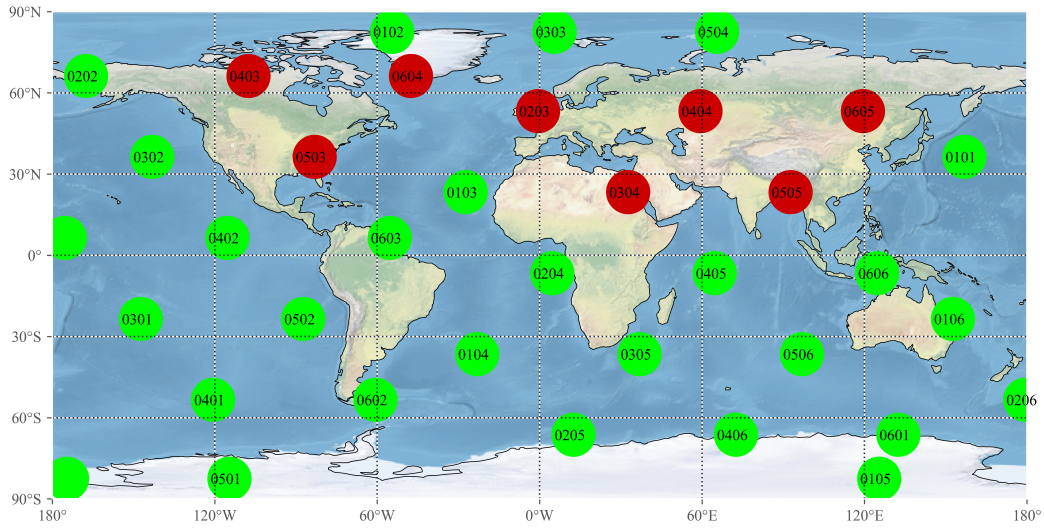

Figure 5. Constellation 1 snapshot.

5.1.3. Communication Parameter

Communication parameters mainly include the transmission rate, transmission power, etc., which are summarized in Table 4. These parameters are referenced in [11].

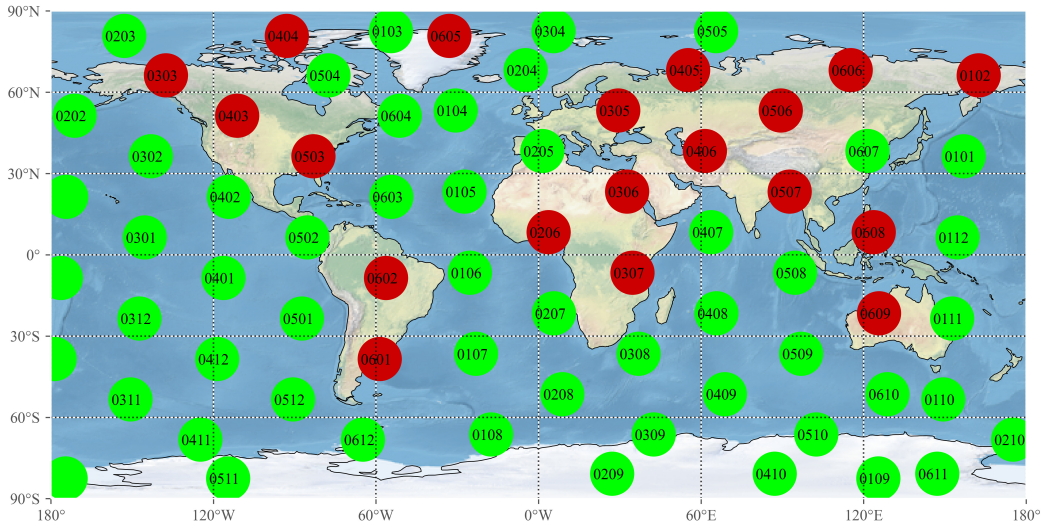

Figure 6. Constellation 2 snapshot.

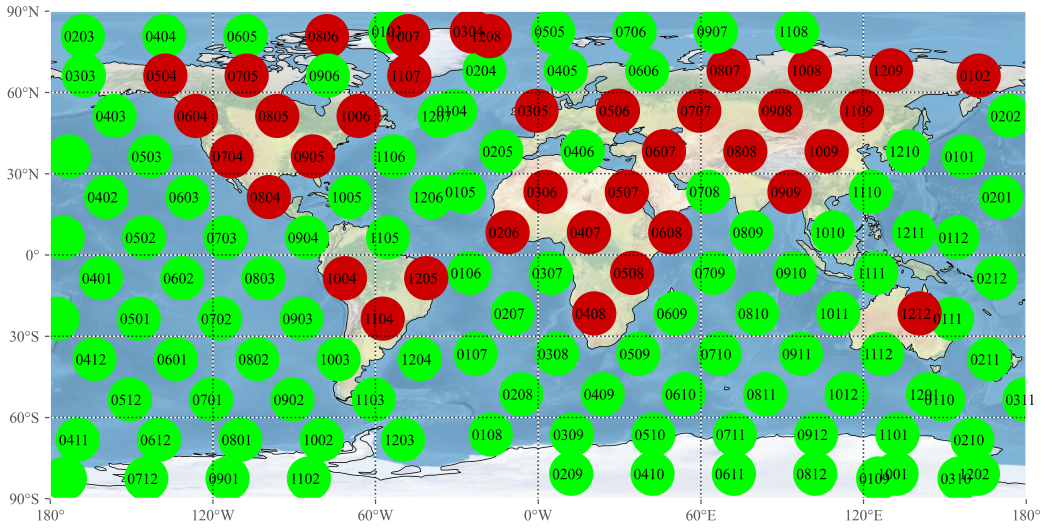

Figure 7. Constellation 3 snapshot.

Table 4. Communication parameters.

Parameter	Value
Satellite terminal uplink power	0.2 W
Satellite terminal transmit rate	2 Mbps
Inter-satellite transmit power	3 W
Inter-satellite transmit rate	30 Mbps
Boltzmann's constraint K_b	228.6 dB
System noise temperature T_s	25 dB

5.2. Performance Metrics

The comparative performance metrics include satellite resource utilization, task response rate and joint optimization cost. Specifically, satellite resource utilization is defined as the ratio of allocated satellite resources to the total computing capacity of the entire constellation, which is calculated as (17).

$$\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} C_n x_{n,m} / \sum_{m\in\mathcal{M}} F_m \tag{17}$$

The task response rate is defined as the ratio of the number of successfully offloaded tasks to the number of all current task offloading requests, which can reflect the ratio that user requests can be accessed. In addition, considering there is little research on the whole constellation related offloading strategy, we adopted the ground commonly used two kinds of offloading strategy in edge computing as a benchmark.

(1) Random strategy: When the task offloading request arrives, the orchestrator randomly selects service satellites for the task. If the service satellite has enough computing resources, the task will be accepted; otherwise, the task will be rejected.

(2) First fit strategy: The orchestrator will select service satellites from all the satellites that can be offloaded for the task. The selection is generally randomized. Here we use a greedy strategy to select the satellite with the most resources available.

In the whole constellation offloading scenario, we compared and evaluated the joint optimization strategy with the benchmark strategy with different optimizing options, including energy cost, load balancing cost and joint optimization cost.

5.3. Simulation Results

Figure 8 depicts the satellite resource max utilization under three different offloading strategies with different constellation sizes. From the simulation results, we can find that the utilization of satellite computing resources is only about 20% using the access satellite offloading strategy, and no more than 50% using the neighbor offloading strategy, and the joint optimization offloading strategy can make full use of the satellite computing resources of the whole system to improve the utilization efficiency of the constellation edge cloud network system.

For example, constellation 2 with 72 satellites, as shown in Figure 9, under access satellite offloading strategy, the satellite resource utilization with each concurrent tasks demand varies slightly in fluctuations. As the number of offloading tasks increases, the whole constellation satellite resource utilization is maintained at no more than 20%. The neighbor satellite offloading strategy utilizes the neighbor node of the satellite directly connected to the satellite terminal, which has a higher resource utilization ratio than the access satellite strategy, but it does not increase after exceeding the computing capacity of all the neighboring satellites. In practice, the more imbalanced distribution of tasks will lead to lower utilization of satellite resources. Figure 10 shows the task response rate under the constellation 3 of 144 satellites with different concurrent tasks. It can be seen that as the number of tasks increases, the task response rate decreases gradually under the offloading strategy of access satellites and neighbor satellites, and the all-constellation joint offloading strategy can meet the access of more offloading tasks. In the scenario of whole-constellation offloading, it can be seen from Figure 11 that the satellite resources utilization rate under the joint optimization strategy is the highest, which is 40% higher than the random offloading strategy and about 20% higher than the FirstFit strategy. When the computational capacity of the whole constellation meets the requirements of all offloading tasks, the joint optimization strategy can maintain the task response rate of 100%, while the random strategy and FirstFit strategy both begin to decline when the number of concurrent tasks reaches a certain value, as shown in Figure 12. In constellation 2 with 72 satellite scenarios, Figure 13 shows the three offloading strategies of energy cost. It is obvious that the joint optimization offloading strategy has the lowest energy costs, and the first fit strategy only considered the calculation of energy cost, and ignored the optimal choice of the path, causing the total energy cost at the highest levels. Figure 14 shows the load balancing costs of the three offloading strategies under 72 satellites. The joint optimization strategy has the lowest load cost because the load balancing level is an important optimization item in the joint optimization. The first fit strategy greedily selects the satellite node with the most remaining resources each time, which also has a certain load balancing effect. The random strategy does not consider the load between satellite resources, and the load balancing cost is the highest. From the simulation results of Figure 15, we know that the joint cost of the joint optimization strategy is the lowest

in the three whole constellation offloading strategies, and the joint cost of the random strategy is higher than the first fit strategy. This is because the weight coefficient β of the objective function is set to 5, which causes load balancing to affect their joint cost. However, no matter how β changes, the cost of the joint optimization strategy will remain the lowest. Finally, we reviewed the five offloading strategies, including access satellite offloading, neighbor satellite offloading, random offloading, first fit offloading and joint offloading, as shown in Figure 16. Compared with other offloading strategies, the joint optimization strategy of the whole LEO constellation has the best performance, and the joint optimization strategy not only considers the energy consumption generated by the computation, but also considers the transmission path energy loss and the load balancing between the satellites. Furthermore, we can tune β to adjust the trade-off between energy cost and load balancing cost.

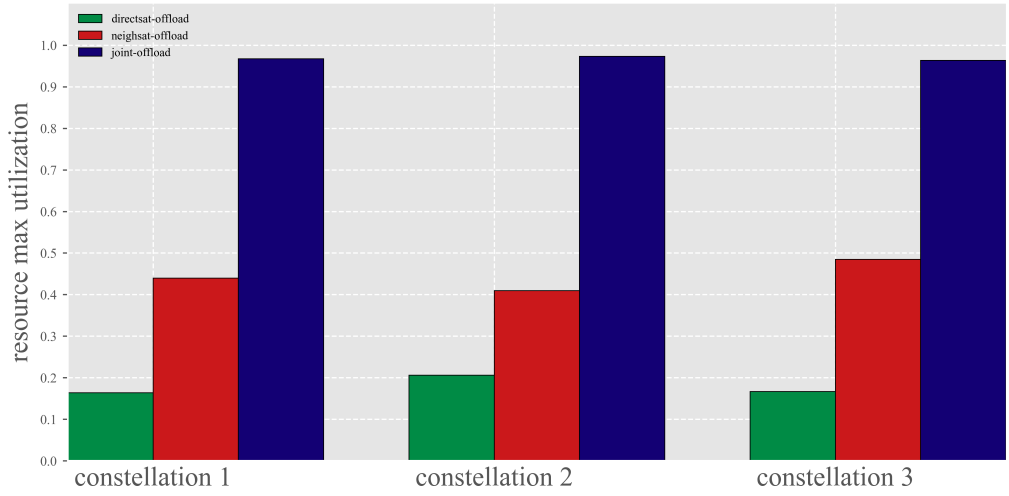

Figure 8. The maximum utilization of satellites computing resources in different LEO constellation sizes.

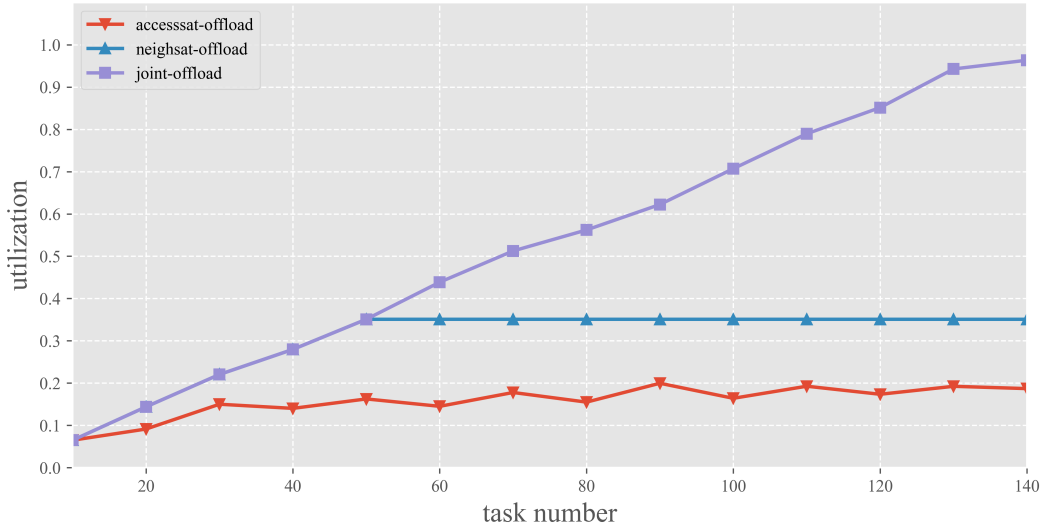

Figure 9. Satellite computing resource utilization varies with number of tasks in constellation 2 with 72 satellites.

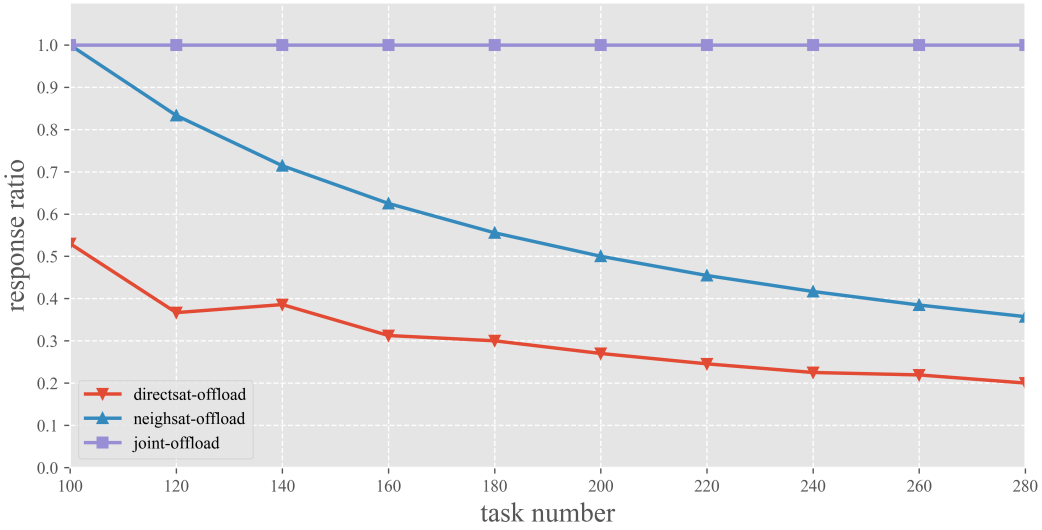

Figure 10. Task response rate varies with number of tasks in constellation 3 with 144 satellites.

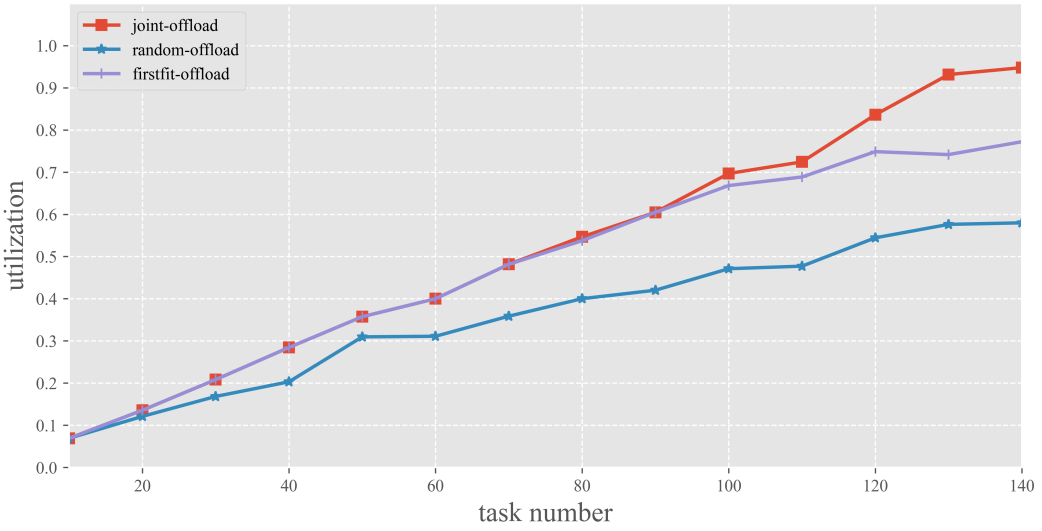

Figure 11. Satellite computing resource utilization of the three offloading strategies varies with the number of tasks in the whole constellation offloading scenario.

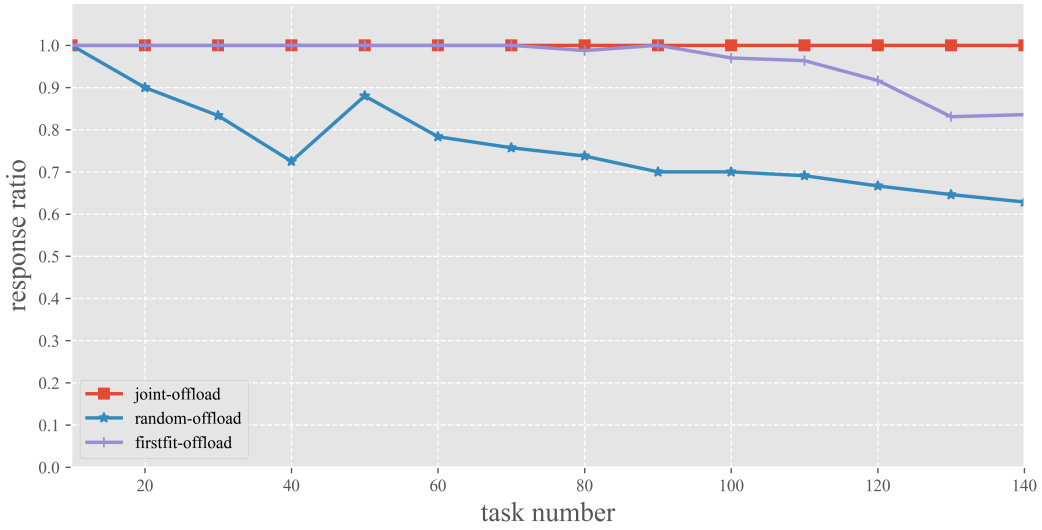

Figure 12. Task response rate of the three offloading strategies varies with the number of tasks in the whole constellation offloading scenario.

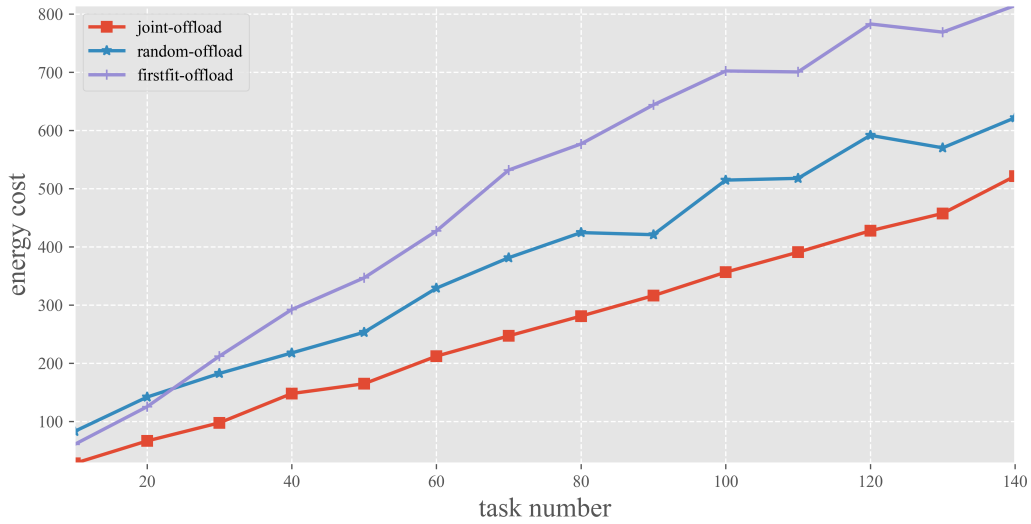

Figure 13. Energy cost of the three offloading strategies varies with the number of tasks in the whole constellation offloading scenario.

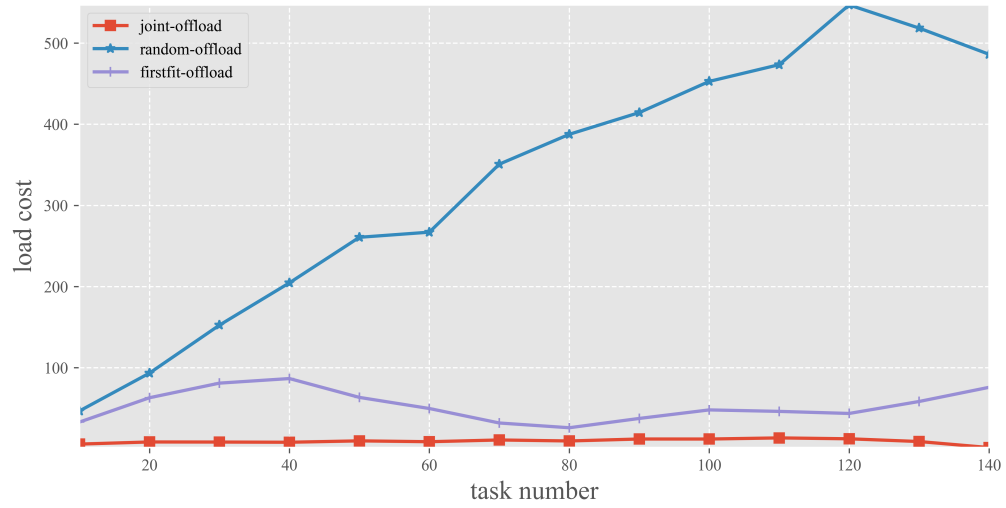

Figure 14. Load balance cost of the three offloading strategies varies with the number of tasks in the whole constellation offloading scenario.

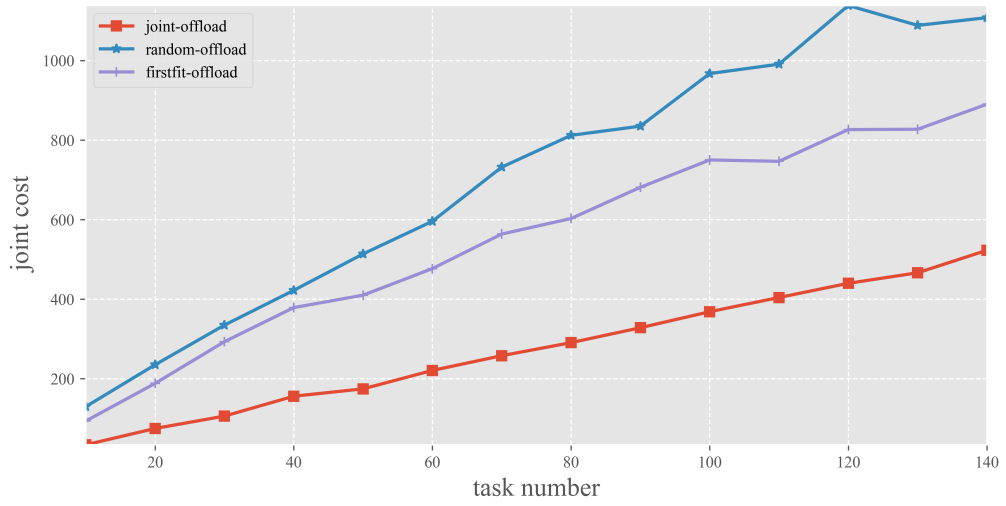

Figure 15. Joint optimization cost of the three offloading strategies varies with the number of tasks in the whole constellation offloading scenario.

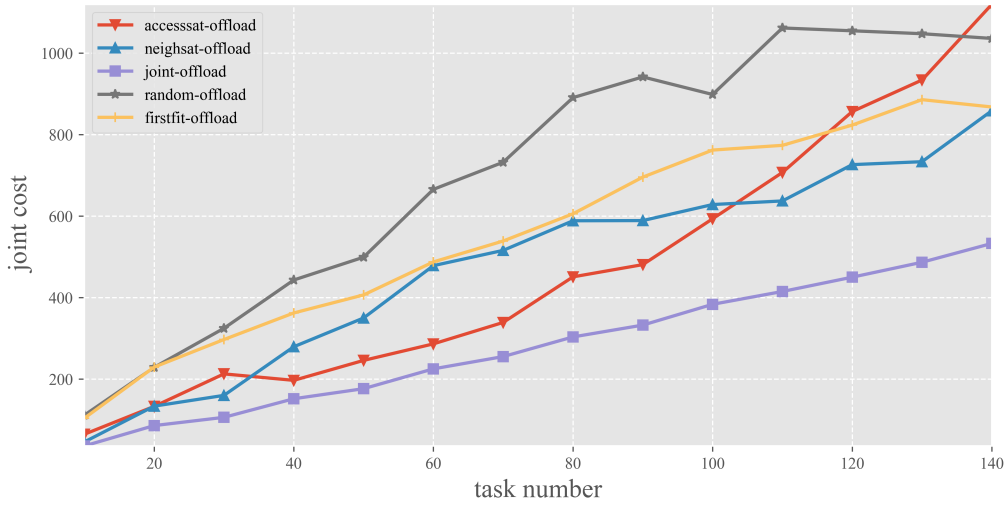

Figure 16. Joint optimization cost of all offloading strategies varies with the number of tasks.

6. Conclusions

In this paper, we proposed the concept of the LEO constellation edge cloud network (LEO-ECN), established the system model of LEO-ECN, and gave the offloading mechanism of LEO-ECN. In this model, the computational offloading problem of energy consumption and load balancing combined optimization is studied from the system point of view. Especially for energy consumption modeling, considering the characteristics of multi-hop network of low orbit constellation, the energy consumption cost model combining computing and network is given, and the load balance of the whole constellation computing resources is also considered. Finally, we solved the problems, and conducted a series of simulation experiments. In different constellation scales, the proposed offloading strategy was compared with access satellite offloading, neighbor satellite offloading, random offloading and first fit offloading. The experimental results showed that our proposed offloading strategy has higher satellite resource utilization and better performance.

Author Contributions: Conceptualization, C.S., Y.Z. and F.D.; methodology, T.H. and F.D.; software and validation, F.D.; resources, T.H. and F.D.; data curation, C.L.;writing original draft preparation, F.D.; supervision, C.S.; project administration, Y.Z.; funding acquisition, T.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant No. 62171046.

Data Availability Statement: The data presented in this research are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LEO	Low Earth Orbit
IoT	Internet of Thing
LEO-ECN	LEO Constellation Edge Cloud Network
5G	5th Generation Mobile Networks
6G	6th Generation Mobile Networks
VR/AR	Virtual Reality/Augmented Reality
ADMM	Alternating Direction Method of Multipliers
TDMA	Time Division Multiple Access
MEC	Mobile Edge Computing
CPU	Central Processing Unit

References

1. Butash, T.; Garland, P.; Evans, B. Non-geostationary Satellite Orbit Communications Satellite Constellations History. *Int. J. Satell. Commun. Netw.* **2020**, *39*, sat.1375. [CrossRef]
2. Starlink. Available online: https://en.wikipedia.org/wiki/Starlink#cite_note-JM-stats-1 (accessed on 18 May 2022).
3. Bhattacherjee, D.; Kassing, S.; Licciardello, M.; Singla, A. In-Orbit Computing: An Outlandish Thought Experiment? In Proceedings of the 19th ACM Workshop on Hot Topics in Networks, Virtual Event, 4–6 November 2020; ACM: New York, NY, USA, 2020; pp. 197–204. [CrossRef]
4. Wang, Y.; Yang, J.; Guo, X.; Qu, Z. A Game-Theoretic Approach to Computation Offloading in Satellite Edge Computing. *IEEE Access* **2020**, *8*, 12510–12520. [CrossRef]
5. Xie, R.; Tang, Q.; Wang, Q.; Liu, X.; Yu, F.R.; Huang, T. Satellite-Terrestrial Integrated Edge Computing Networks: Architecture, Challenges, and Open Issues. *IEEE Netw.* **2020**, *34*, 224–231. [CrossRef]
6. Wang, B.; Feng, T.; Huang, D. A Joint Computation Offloading and Resource Allocation Strategy for LEO Satellite Edge Computing System. In Proceedings of the 2020 IEEE 20th International Conference on Communication Technology (ICCT), Nanning, China, 28–31 October 2020; pp. 649–655. [CrossRef]
7. Wang, Y.; Zhang, J.; Zhang, X.; Wang, P.; Liu, L. A Computation Offloading Strategy in Satellite Terrestrial Networks with Double Edge Computing. In Proceedings of the 2018 IEEE International Conference on Communication Systems (ICCS), Chengdu, China, 19–21 December 2018; pp. 450–455. [CrossRef]
8. Tang, Q.; Fei, Z.; Li, B.; Han, Z. Computation Offloading in LEO Satellite Networks With Hybrid Cloud and Edge Computing. *IEEE Internet Things J.* **2021**, *8*, 9164–9176. [CrossRef]
9. He, M.; Zhong, L.; Tan, H.; Qu, Y.; Lai, J. A Novel Edge Computing Server Selection Strategy of LEO Constellation Broadband Network. In Proceedings of the 2020 IEEE World Congress on Services (SERVICES), Beijing, China, 18–23 October 2020; pp. 275–280. [CrossRef]
10. Chen, M.; Chai, R.; Chen, Q. Joint Route Selection and Resource Allocation Algorithm for Data Relay Satellite Systems Based on Energy Efficiency Optimization. In Proceedings of the 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP), Xi'an, China, 23–25 October 2019; pp. 1–6. [CrossRef]
11. Golkar, A.; Lluch i Cruz, I. The Federated Satellite Systems Paradigm: Concept and Business Case Evaluation. *Acta Astronaut.* **2015**, *111*, 230–248. [CrossRef]
12. Werner, M. A Dynamic Routing Concept for ATM-based Satellite Personal Communication Networks. *IEEE J. Sel. Areas Commun.* **1997**, *15*, 1636–1648. [CrossRef]
13. Chen, X. Decentralized Computation Offloading Game for Mobile Cloud Computing. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 974–983. [CrossRef]
14. Diamond, S.; Boyd, S. CVXPY: A Python-embedded Modeling Language for Convex Optimization. *J. Mach. Learn. Res.* **2016**, *17*, 2909–2913.
15. Satellite Constellation Visualization. Available online: https://savi.sourceforge.io/ (accessed on 18 May 2022).
16. Met Office. *Cartopy: A Cartographic Python Library with a Matplotlib Interface*; Met Office: Exeter, UK, 2010.